(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 August 2017 (31.08.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/147257 A1

(51) International Patent Classification:
G06F 3/06 (2006.01)

(21) International Application Number:
PCT/US2017/019067

(22) International Filing Date:
23 February 2017 (23.02.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15/052,246 24 February 2016 (24.02.2016) US

(71) Applicant: **NETAPP, INC.** [US/US]; 495 East Java Drive, Suite 500, Sunnyvale, California 95089 (US).

(72) Inventors: **CALLAHAN, Travis Ian**; 308 Rushing Wind Way, Apex, North Carolina 27502 (US). **BHATTACHARJEE, Mrinal K.**; 201, Chaitanya Residency, 3rd Cross, Wind Tunnel Road, Murugeshpalya, Bangalor 560017 (IN). **KONDAPALLI, Ganga Bhavani**; 495 East Java Drive, Bldg. 1, Sunnyvale, California 95089 (US). **GUTKNECHT, William Arthur**; 4402 Shaw Farm Circle, Greensboro, North Carolina 27406 (US). **ZHU, Andrew**; 103 Filtrona Court, Cary, North Carolina 27519 (US).

(74) Agent: **SHAMI, Khaled**; Nixon Peabody LLP, 799 9th Street, NW, Washington, District of Columbia 20001-4501 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: ADAPTIVE, SELF LEARNING CONSISTENCY POINT TRIGGERS

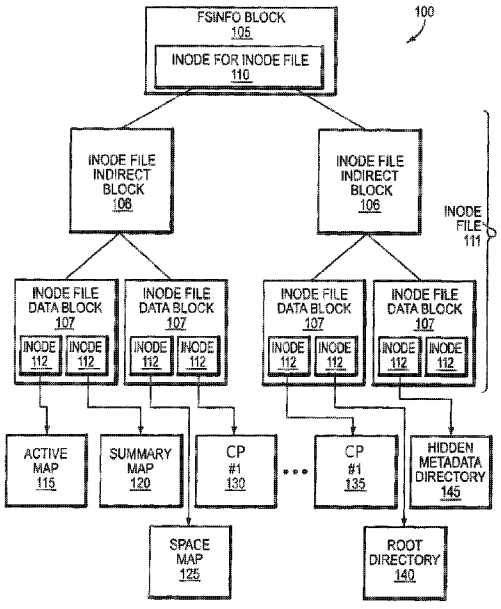


FIG. 1

(57) Abstract: Presented herein are methods, non-transitory computer readable media, and devices for allocating resources to a particular volume and triggering a consistency point based on the amount given to each volume, thus providing dynamic consistency point triggers. Methods for providing dynamic consistency point triggers are disclosed which include: determining a volume's capacity to undertake the resources based on the volume's performance; receiving a portion of the divided resources based on total system resources available within the storage server and the volume's performance; and triggering a consistency point upon exhausting a threshold percentage of the received portion of resources.

WO 2017/147257 A1

Published:

— *with international search report (Art. 21(3))*

ADAPTIVE, SELF LEARNING CONSISTENCY POINT TRIGGERS

RELATED APPLICATIONS

[0001] In full or in part, this application describes the same method and device as presented in co-pending United States utility patent application number 15/052,246, filed on February 24, 2016, and claims international date priority thereof as a Patent Cooperation Treaty application. The subject matter of which is hereby incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The present disclosure relates generally to mass data storage networks and data storage filing systems. More particularly, aspects of this disclosure relate to methods, non-transitory computer readable media, and devices for providing dynamic consistency point triggers.

BACKGROUND

[0003] A storage system typically comprises one or more storage devices into which information may be entered, and from which information may be obtained, as desired. The storage system includes a storage operating system that functionally organizes the system by, inter alia, invoking storage operations in support of a storage service implemented by the system. The storage system may be implemented in accordance with a variety of storage architectures including, but not limited to, a network-attached storage environment, a storage area network and a disk assembly directly attached to a client or host computer. The storage devices are typically disk drives organized as a disk array, is wherein the term "disk" commonly describes a self-contained rotating magnetic media storage device. The term disk in this context is synonymous with hard disk drive (HDD) or direct access storage device (DASD).

[0004] Storage of information on the disk array is preferably implemented as one or more storage “volumes” of physical disks, defining an overall logical arrangement of disk space. The disks within a volume are typically organized as one or more groups, wherein each group may be operated as a Redundant Array of Independent (or Inexpensive) Disks (RAID). Most RAID implementations enhance the reliability/integrity of data storage through the redundant writing of data "stripes" across a given number of physical disks in the RAID group, and the appropriate storing of redundant information (parity) with respect to the striped data. The physical disks of each RAID group may include disks configured to store striped data (ie., data disks) and disks

configured to store parity for the data (ie., parity disks). The parity may thereafter be retrieved to enable recovery of data lost when a disk fails.

[0005] The storage operating system of the storage system may implement a high-level module, such as a file system, to logically organize the information stored on the disks as a hierarchical structure of directories, files and blocks. For example, each "on-disk" file may be implemented as set of data structures, i.e., disk blocks, configured to store information, such as the actual data for the file. These data blocks are organized within a volume block number (vbn) space that is maintained by the file system. The file system organizes the data blocks within the vbn space as a "logical volume"; each logical volume may be, although is not necessarily, associated with its own file system. The file system typically consists of a contiguous range of vbns from zero to n, for a file system of size n-1 blocks.

[0006] A known type of file system is a write-anywhere file system that does not overwrite data on disks. If a data block is retrieved (read) from disk into a memory of the storage system and "dirtied" (i.e., updated or modified) with new data, the data block is thereafter stored (written) to a new location on disk to optimize write performance. A write-anywhere file system may initially assume an optimal layout such that the data is substantially contiguously arranged on disks. The optimal disk layout results in efficient access operations, particularly for sequential read operations, directed to the disks.

[0007] The storage operating system may further implement a storage module, such as a RAID system, that manages the storage and retrieval of the information to and from the disks in accordance with input/output (I/O) operations. The RAID system is also responsible for parity operations in the storage system. Note that the file system only "sees" the data disks within its vbn space; the parity disks are "hidden" from the file system and, thus, are only visible to the RAID system. The RAID system typically organizes the RAID groups into one large "physical" disk (i.e., a physical volume), such that the disk blocks are concatenated across all disks of all RAID groups. The logical volume maintained by the file system is then "disposed over" the physical volume maintained by the RAID system.

[0008] The storage system may be configured to operate according to a client/server model of information delivery to thereby allow many clients to access the directories, files and blocks stored on the system. In this model, the client may comprise an application, such as a database application, executing on a computer that "connects" to the storage system over a computer network, such as a point-to-point link, shared local area network, wide area network or virtual private network implemented over a public network, such as the Internet. Each client may request the services of the file system by issuing file system protocol messages (in the form of

packets) to the storage system over the network. By supporting a plurality of file system protocols, such as the conventional Common Internet File System (CIFS) and the Network File System (NFS) protocols, the utility of the storage system is enhanced.

[0009] When accessing a block of a file in response to servicing a client request, the file system specifies a vbn that is translated at the file system/RAID system boundary into a disk block number (dbn) location on a particular disk (disk, dbn) within a RAID group of the physical volume. Each block in the vbn space and in the dbn space is typically fixed, e.g., 4 k bytes (KB), in size; accordingly, there is typically a one-to-one mapping between the information stored on the disks in the dbn space and the information organized by the file system in the vbn space. The (disk, dbn) location specified by the RAID system is further translated by a disk driver system of the storage operating system into a sector (or similar granularity) on the specified disk.

[0010] The requested block is then retrieved from disk and stored in a buffer cache of the memory as part of a buffer tree of the file. The buffer tree is an internal representation of blocks for a file stored in the buffer cache and maintained by the file system. Broadly stated, the buffer tree has an inode at the root (top-level) of the file. An inode is a data structure used to store information, such as metadata, about a file, whereas the data blocks are structures used to store the actual data for the file. The information contained in an inode may include, e.g., ownership of the file, access permission for the file, size of the file, file type and references to locations on disk of the data blocks for the file. The references to the locations of the file data are provided by pointers, which may further reference indirect blocks that, in turn, reference the data blocks, depending upon the quantity of data in the file. Each pointer may be embodied as a vbn to facilitate efficiency among the file system and the RAID system when accessing the data on disks.

[0011] The file system, such as the write-anywhere file system, maintains information about the geometry of the underlying physical disks (e.g., the number of blocks in each disk) in the storage system. The RAID system provides the disk geometry information to the file system for use when creating and maintaining the vbn-to-disk, dbn mappings used to perform write allocation operations. The file system maintains block allocation data structures, such as an active map, a space map, a summary map and snapmaps. These mapping data structures describe which blocks are currently in use and which are available for use and are used by a write allocator of the file system as existing infrastructure for the logical volume.

[0012] Specifically, the snapmap denotes a bitmap file describing which blocks are used by a consistency point. The write-anywhere file system has the capability to generate a consistency point of its active file system. An “active file system” is a file system to which data can be both

written and read or, more generally, an active store that responds to both read and write I/O operations. A consistency point is a space conservative, point-in-time read-only image of data accessible by name that provides a consistent image of that data (such as a storage system) at some previous time. More particularly, a consistency point is a point-in-time representation of a storage element, such as an active file system, file or database, stored on a storage device (e.g., on disk) or other persistent memory and having a name or other identifier that distinguishes it from other consistency point taken at other points in time. A consistency point can also include other information (metadata) about the active file system at the particular point in time for which the image is taken.

[0013] The write-anywhere file system supports multiple consistency points that are generally created on a regular schedule. Each consistency point refers to a copy of the file system that diverges from the active file system over time as the active file system is modified. Each consistency point is a restorable version of the storage element (e.g., the active file system) created at a predetermined point in time and, as noted, is "read-only" accessible and "space-conservative." Space conservative denotes that common parts of the storage element in multiple consistency points share the same file system blocks. Only the differences among these various consistency points require extra storage blocks. The multiple consistency points of a storage element are not independent copies, each consuming disk space; therefore, creation of a consistency point on the file system is instantaneous, since no entity data needs to be copied. Read-only accessibility denotes that a consistency point cannot be modified because it is closely coupled to a single writable image in the active file system. The closely coupled association between a file in the active file system and the same file in a consistency point obviates the use of multiple "same" files.

[0014] The active map denotes a bitmap file describing which blocks are used by the active file system. As described prior, a consistency point may contain metadata describing the file system as it existed at the point in time that the image was taken. In particular, a consistency point captures the active map as it existed at the time of consistency point creation; this file is also known as the snapmap for the consistency point. Note then that a snapmap denotes a bitmap file describing which blocks are used by a consistency point. The summary map denotes a file that is an inclusive logical OR bitmap of all snapmaps. By examining the active and summary maps, the file system can determine whether a block is in use by either the active file system or any consistency point. The space map denotes a file including an array of numbers that describe the number of storage blocks used in a block allocation area. In other words, the space map is

essentially a logical OR bitmap between the active and summary maps to provide a condensed version of available "free block" areas within the vbn space.

[0015] FIG. 1 is a schematic block diagram of an exemplary on-disk storage structure 100 of a logical volume of a storage system. As noted, a logical volume is typically associated with a file system and comprises data blocks organized within a vbn space. Each logical volume (hereinafter "volume") has a file system information (fsinfo) block that is preferably stored at a fixed location within, e.g., a RAID group. Fsinfo block 105 is the root of the on-disk storage structure 100, illustratively at vbns 1 and 2. When loading the volume, the storage operating system accesses those vbns to acquire the fsinfo block 105.

[0016] The fsinfo block 105 includes a variety of metadata that describes the state of the file system; also included in the fsinfo block 105 is an inode for an inode file 110. All inodes of the write-anywhere file system are organized into the inode file 111. Like any other file, the inode of the inode file is the root of the buffer tree that describes the location of blocks of the file. As such, the inode of the inode file may directly reference (point to) data blocks 107 of the inode file 111 or may reference indirect blocks 106 of the inode file 111 that, in turn, reference data blocks of the inode file. In this example, the inode for the inode file 110 includes an exemplary buffer tree comprising a plurality of inode file indirect blocks 106 that, in turn, point to inode file data blocks 107. Within each data block of the inode file are inodes 112, each of which serves as the root of a file. Among the inodes of the inode file 110, there are inodes for special metadata files, such as an active map 115, a summary map 120, a space map 125, a root directory 140 and a metadata directory 145. All user files in the file system are organized under the root directory 140, while various metadata files associated with the file system are stored under the metadata directory 145.

[0017] The inode file may further include inodes that reference a plurality of consistency points 130, 135. These consistency point inodes are the root level inodes of consistency points of the active file system. Each volume has special reserved inode numbers within its vbn space; a plurality of those inode numbers (e.g., 31) is reserved for consistency points. When a consistency point is generated of the active file system, a copy of the inode for the inode file is generated (hereinafter the "consistency point root") and assigned one of the reserved consistency point inode numbers. Thus, to access a consistency point at a particular point in time, the storage operating system accesses the appropriate consistency point root of the consistency point.

[0018] With system wide static consistency points, the system is incapable of preventing one volume from consuming all or most of the resources leaving none for other volumes. When this occurs and the volume is unable to release buffers and non-volatile logs (NVLogs) in a timely

manner, operations for that volume experiences latency due to lack of resources. Further, other volumes experience operational latency due to the behavior of this volume. Adding to the unpredictable nature of consistency points are the storage server internal operations, which may run during the consistency point. These internal workloads generate more work for the running consistency point leading to unpredictable consistency point duration, which translates to inconsistent performance within the storage server. The present disclosure is directed to alleviating this limitation.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] FIG. 1 is a schematic block diagram of an on-disk storage structure of a volume of a storage system.

[0020] FIG. 2 is a schematic block diagram of an environment including a storage system that may be added advantageously used with the present disclosure.

[0021] FIG. 3 is a schematic block diagram of the array that may be advantageously used with the present disclosure.

[0022] FIG. 4 is an exemplary method for resource partitioning for a specific volume.

[0023] FIG. 5 is a schematic block diagram of a RAID group that may be advantageously used with the present disclosure.

[0024] FIG. 6 is an exemplary method for allocating resources to a requesting volume.

[0025] The present disclosure is susceptible to various modifications and alternative forms, and some representative embodiments have been shown by way of example in the drawings and will be described in detail herein. It should be understood, however, that the inventive aspects are not limited to the particular forms illustrated in the drawings. Rather, the disclosure is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the disclosure as defined by the appended claims.

DETAILED DESCRIPTION OF ILLUSTRATED EXAMPLES

[0026] The present disclosure is directed to a system and method providing dynamic consistency point triggers. As used herein, the term "mass data storage system" generally refers to the computer-executable code operable on a computer to perform a storage function that manages data access and may, in the case of a storage system, implement data access semantics of a general purpose operating system. The mass data storage system can also be implemented as a microkernel, an application program operating over a general-purpose operating system, such

as UNIX® or Windows NT®, or as a general-purpose operating system with configurable functionality, which is configured for storage applications as described herein.

[0027] In addition, it will be understood to those skilled in the art that the disclosure described herein may apply to any type of special-purpose (e.g., file server, filer or storage serving appliance) or general-purpose computer, including a standalone computer or portion thereof (i.e. a workload), embodied as or including a storage system. Moreover, the teachings of this disclosure can be adapted to a variety of storage system architectures including, but not limited to, a network-attached storage environment, a storage area network, a disk assembly directly-attached to a client or host computer and, illustratively, a cluster of interconnected storage system nodes. The term "storage system" should therefore be taken broadly to include such arrangements in addition to any subsystems configured to perform a storage function and associated with other equipment or systems. It should be noted that while this description is written generally in terms of a log-structured file system, the teachings of the present disclosure may be utilized with any suitable file system, including a write anywhere file system.

[0028] It is desirable to improve the performance of storage servers, and one way to do so is by providing dynamic consistency point triggers. Prior to the current disclosure, consistency points are triggered for every dirty logical volume (hereinafter "volume") in the system thus the performance of each volume directly affects all other volumes. With this disclosure, a level of isolation between the volumes can be provided by dividing resources between the volumes and allowing the volumes to take consistency points as their resources become exhausted. Aspects of this disclosure are directed to methods, non-transitory computer readable media, and devices for providing consistency point triggers that dynamically adjust to system conditions, for example, the changing throughput of a raid group or the incoming workload to a volume. These consistency point triggers are designed to make a volume's consistency point duration more predictable and ensure all volumes have the resources needed to continue serving data. In an exemplary steady state, consistency points are triggered for an individual volume based on the resource consumption of the individual volume.

[0029] In one embodiment, resource-partitioning model is the basis of the consistency point trigger design. The partitioning is applied to dirty buffers and the non-volatile log of a storage server, as those are the main resources consumed by write anywhere file system operations that are released by the consistency point. However, the partition model could be extended to include additional resources if necessary. In one embodiment, to partition managed resources, each volumes resource fill rate (i.e. consumption rate) and resource drain rate (i.e. rate of release) is used as an exemplary method of resource partitioning. This allows the system to restrict the

resource usage of a volume that is overloaded (i.e. fill rate > drain rate) in favor of a healthy volume. By applying these principles the partitioning model has built in support for using a volume's actual write throughput and provides a soft resource guarantee to all volumes. Disclosed, for example, each volume receives a portion of dirty buffers based on its contribution to the total system fill rate. The resource amount given to each volume then drives when a consistency point is triggered for the volume.

[0030] In another embodiment, internal workloads that generate work for consistency points are dynamically detected and the amount of work each is allowed to generate is restricted. When the storage server attempts to dirty an in-memory buffer, a set of rules is applied to the message, which determines if the running message is an internal workload message. If the criterion is met the message type is labeled as an internal workload message type. This information is then used when processing new messages to quickly identify an internal workload message. Each buffer dirtied by an internal workload message is accounted for in order to stop new work from being generated when a defined limit is reached, or once the consistency point completes.

[0031] Referring to FIG. 2, a schematic block diagram of an environment 200 including a storage system 220 that may be advantageously used with the present disclosure. The storage system is a computer that provides storage service relating to the organization of information on storage devices, such as disks 230 of a disk array 260. The storage system 220 comprises a processor 222, a memory 224, a network adapter 226 and a storage adapter 228 interconnected by a system bus 225. The storage system 220 also includes a storage operating system 300 that preferably implements a high-level module, such as a file system, to logically organize the information as a hierarchical structure of directories, files and special types of files called virtual disks (hereinafter "blocks") on the disks.

[0032] In the illustrative embodiment, the memory 224 comprises storage locations that are addressable by the processor and adapters for storing software program code. A portion of the memory may be further organized as a "buffer cache" 270 for storing certain data structures associated with the present disclosure. The processor and adapters may, in turn, comprise processing elements and/or logic circuitry configured to execute the software code and manipulate the data structures. Storage operating system 300, portions of which are typically resident in memory and executed by the processing elements, functionally organizes the system 220 by, inter alia, invoking storage operations executed by the storage system. It will be apparent to those skilled in the art that other processing and memory means, including various computer readable media, may be used for storing and executing program instructions pertaining to the inventive technique described herein.

[0033] The network adapter 226 comprises the mechanical, electrical and signaling circuitry needed to connect the storage system 220 to a client 210 over a computer network 240, which may comprise a point-to-point connection or a shared medium, such as a local area network. Illustratively, the computer network 240 may be embodied as an Ethernet network or a Fibre Channel (FC) network. The client 210 may communicate with the storage system over network 240 by exchanging discrete frames or packets of data according to pre-defined protocols, such as the Transmission Control Protocol/Internet Protocol (TCP/IP).

[0034] The client 210 may be a general-purpose computer configured to execute applications 212. Moreover, the client 210 may interact with the storage system 220 in accordance with a client/server model of information delivery. That is, the client may request the services of the storage system, and the system may return the results of the services requested by the client, by exchanging packets 250 over the network 240. The clients may issue packets including file-based access protocols, such as the Common Internet File System (CIFS) protocol or Network File System (NFS) protocol, over TCP/IP when accessing information in the form of files and directories. Alternatively, the client may issue packets including block-based access protocols, such as the Small Computer Systems Interface (SCSI) protocol encapsulated over TCP (iSCSI) and SCSI encapsulated over Fibre Channel (FCP), when accessing information in the form of blocks.

[0035] The storage adapter 228 cooperates with the storage operating system 300 executing on the system 220 to access information requested by a user (or client). The information may be stored on any type of attached array of writable storage device media such as video tape, optical, DVD, magnetic tape, bubble memory, electronic random access memory, micro-electro mechanical and any other similar media adapted to store information, including data and parity information. However, as illustratively described herein, the information is preferably stored on the disks 230, such as HDD and/or DASD, of array 260. The storage adapter includes input/output (I/O) interface circuitry that couples to the disks over an I/O interconnect arrangement, such as a conventional high-performance, FC serial link topology.

[0036] Storage of information on array 260 can be implemented as one or more storage "volumes" that comprise a collection of physical storage disks 230 cooperating to define an overall logical arrangement of volume block number (vbn) space on the volume(s). Each logical volume is generally, although not necessarily, associated with its own file system. The disks within a logical volume/file system are typically organized as one or more groups, wherein each group may be operated as a Redundant Array of Independent (or Inexpensive) Disks (RAID). Most RAID implementations, such as a RAID-4 level implementation, enhance the

reliability/integrity of data storage through the redundant writing of data "stripes" across a given number of physical disks in the RAID group, and the appropriate storing of parity information with respect to the striped data. An illustrative example of a RAID implementation is a RAID-4 level implementation, although it should be understood that other types and levels of RAID implementations may be used in accordance with the inventive principles described herein.

[0037] To facilitate access to the disks 230, the storage operating system 300 implements a write-anywhere file system that cooperates with virtualization modules to "virtualize" the storage space provided by disks 230. The file system logically organizes the information as a hierarchical structure of named directories and files on the disks. Each "ondisk" file may be implemented as set of disk blocks configured to store information, such as data, whereas the directory may be implemented as a specially formatted file in which names and links to other files and directories are stored. The virtualization modules allow the file system to further logically organize information as a hierarchical structure of blocks on the disks that are exported as named logical unit numbers (luns).

[0038] FIG. 3 is a schematic block diagram of the array 260 that may be advantageously used with the present disclosure. The volumes can be isolated by dividing resources 330 between the volumes 350 and allowing the volumes to take consistency points as their resources are exhausted. The array 260 can include one or more storage volumes (e.g., VOL1 & VOL2 350) that include a cluster of physical storage disks 330, defining an overall logical arrangement of disk space. Specifically, each volume 350 is constructed from an array of physical disks 330 that are organized as RAID groups 340, 342, and 344. Each volume 350 receives a portion of dirty buffers based on its contribution to the total system fill rate via the resource partition. The resource partition is the total amount of a given resource pre-allocated to a volume. This is the sum of the rate partition and the non-rate partition of a given volume. The resource amount given to each volume 350 then drives when a consistency point is triggered for the volume 350. Fill rate is the consumption rate of a given resource, NVLog or dirty buffers, by a volume 350. NVLogs are configured to temporarily record ("log") I/O requests, such as write requests. The drain rate is the rate a volume returns a resource, NVLog or dirty buffers. This is measured over the duration of a consistency point. The rate partition is the amount of a given resource pre-allocated to a volume based on the fill and drain rate for the corresponding resource.

[0039] An exemplary method for providing dynamic consistency point triggers will now be described with reference to FIGS. 2-3. Referring more specifically to FIG. 4, an exemplary method for resource partitioning for a specific volume. To fairly partition managed resources 330, each volumes 350 resource fill rate (i.e. consumption rate) and resource drain rate (i.e. rate

of release) can be used to determine proper resource partitioning. This allows the system to restrict the resource usage of a volume that is overloaded (i.e. fill rate > drain rate) in favor of a healthy volume. By utilizing these characteristics the partitioning model the disclosure can ensure use of a volume's actual write throughput and provide a soft resource guarantee to all the volumes. The following example illustrates the basic idea of partitioning dirty buffers between volumes based on rates. In method 400, each volume receives a portion of dirty buffers based on its contribution to the total system fill rate. The fill rate is maintained in units of resources per second. The resource amount given to each volume then drives when a consistency point is triggered for the volume. The values discussed below set the volume's dirty buffer suspend limit and the consistency point is triggered on a volume when half of the limit is hit. A file system utilizing consistency points typically halts or otherwise suspends write operations during the time required to perform write allocation during the consistency point. Under heavy loads involving large files, this time may be on the order of tens of seconds, which seriously impedes access latency for clients of the storage system. This is defined as the dirty buffer suspend limit.

[0040] In some embodiments, the resource fill rate is a factor of resource partitioning because it needs to support constant workloads, bursty workloads, and overloaded workloads. To meet these demands more than one fill rate is maintained and special conditions are placed on when these are used. The dirty buffer count used for the fill rate computation is the count of dirty buffers for the next consistency point. The nvlog fill rate count can include numerous sources to cover the various conditions, which impact nvlog usage. The active log size and phantom log sized are summed to obtain the sample nvlog usage. The active log size is the actual nvlog usage of the volume for the next consistency point. The phantom log size is the number of bytes added by operations that don't do actual nvlog consumption for the next consistency point. Phantom nvlog is for an operation to increase the consumed nvlog space of an volume without actually writing data to nvlog. This is used by some features to trigger consistency points based on nvlog usage instead of dirty buffers.

[0041] The summation of active log size and phantom log sized counts forms the nvlog count used to compute the nvlog fill rate. By accounting for these sources the volume is capable of obtaining the nvlog partition required and the system maintains the defined behavior of phantom nvlog. The first fill rate used is one that can handle both bursty and constant workloads. It is sampled and computed for example, every 50 milliseconds based on an exponential moving average with an exemplary smoothing factor of 5.

[0042] A second fill rate maintained is the consistency point fill rate. This covers the resource usage over a consistency point time, which is defined as the difference between the start

time of the current consistency point and the start time of the previous consistency point. The resource usage over this time period is then used to compute a fill rate. The consistency point fill rate is computed at the start of each volume consistency point, not every 50 milliseconds. Obtaining the volume fill rate includes combining the previously mentioned fill rates with additional logic to handle various scenarios in order to produce a usable fill rate. Some special cases include where the fill rate is greater than the drain rate, where there is no change in the resource consumption, there is reduction in resource consumption from a previous sample (e.g. buffer invalidation), and the phantom nvlog reduction messages are suspended for the resource. For example, when the messages are suspended for the resource, the change in resource usage is not an accurate measure of the fill rate. This is due to the burst in messages when the suspended messages are restarted. To handle this case the consistency point fill rate is used to produce a smoother fill rate. Once a volume's resource fill rate is determined it must update the total system resource rate in order for its new rate to be effective. The total fill rate is used to compute the resource rate allocation for the volume during rate partition allocation.

[0043] The resource drain rate captures the ability of a volume to release acquired resources during a consistency point. The drain rate is maintained in units of resource per-second. Unlike the fill rate, the drain rate is computed only at the end of the consistency point. This is because the NVLog is released at the end of the consistency point, and the next consistency point resources can only begin to drain once the running consistency point completes.

[0044] In step 402, the system's dirty buffer suspend limit is determined. In step 404, the fill and drain rate of the individual volumes in the system are determined. In this example, the system's dirty buffer suspend limit is 10,000 dirty buffers. The first volume (Vol1) drain rate, or rate a volume returns a resource over the duration of the consistency point, is 5,000 dirty buffers per second. The second volume (Vol2) drain rate is 10,000 dirty buffers per second. The Vol1 fill rate, or consumption rate of a given resource (i.e., dirty buffers) by a volume, is 1,000 dirty buffers per second. The Vol2 fill rate is 4000 dirty buffers per second. The total system fill rate is 5,000 (fill rate of Vol1 + fill rate of Vol2) dirty buffers per second. These values are shown in the table below.

	Vol Drain Rate (dbufs/sec)	**Vol Fill Rate (dbufs/sec)**
Vol1	5000	1000
Vol2	10000	4000
System Total		5000

[0045] In step 406, the portion of dirty buffers received by each volume can be determined based on the total system's dirty buffer suspend limit, the total system fill rate, the individual volume's fill rate, and the individual volume's drain rate. With the above fill rates, the system dirty buffers suspended would be divided between these volumes as,

$$Volx = \frac{(system's\ dirty\ buffer\ suspend\ limit * volume\ fill\ rate)}{total\ system\ fill\ rate}$$

$$Vol1 = \frac{(10000 * 1000)}{5000} = 2000\ dirty\ buffers$$

$$Vol2 = \frac{(10000 * 4000)}{5000} = 8000\ dirty\ buffers$$

[0046] In step 408, the consistency point is triggered on a volume when a threshold percentage (e.g. 50%) of the limit is hit. These values set the volume's dirty buffer suspend limit and the consistency point is triggered on a volume when half the limit is hit. In this example, given constant rates, the consistency point would trigger on each volume every second (e.g. 10,000 dirty buffers / 5000 dirty buffers per sec). A similar application may be made to nvlog to obtain nvlog resource partitioning.

[0047] As illustrated above, resource partitioning model dynamically distributes resources between volumes based on the fill and drain rate of the individual volumes in the system. For this to achieve consistent behavior the fill rates of the volumes must be known. During certain phases of a volume's life it may not be capable of generating a fill rate that represents the actual needs of the volume. These instances include, but are not limited to, failover, boot, mount/unmount, iron, and idling (i.e. zero workload). To handle these scenarios the resource partitioning needs to have two resource allocation types: rate based allocation of resources; and non-rate based allocation of resources.

[0048] Referring to FIG. 5, a schematic block diagram of a RAID group that may be advantageously used with the present disclosure. Each resource is divided into two pools, a rate pool 502 and a non-rate pool 504. The rate pool 502 is defined as the total number of resource slices minus the resource slices in use by the non-rate pool 504. A resource slice is the minimum amount of a resource allocated to a volume. For example, instead of allocating 1 dirty buffer for

a volume, the volume will be allocated a slice, which is 128 dirty buffers. The default slice size of 128 buffers can be implemented to limit resource waste but large enough to accommodate a few operations preventing resource thrashing. The current default size of 128 buffers can be utilized as it matches the smallest nvlog data bucket allocation. As the value of the slice increases the potential for waste across volumes grows so the selection of a slice size must be done meticulously. With the introduction of the large I/O project (1MB writes) the default slice size can be adjusted to accept large operations.

[0049] The non-rate pool 504 is initially empty; however, it increases in size as slices are removed from the rate pool 502 to satisfy non-rate allocation requests. Note that the sum of the rate allocation and the non-rate allocation must be less than or equal to the total size of the resource. An idle volume (i.e. a volume with zero resource partition) initially allocates the minimum non-idle partition to the non-rate partition of the volume when it transitions to non-idle. This minimum partition stays with the volume while it is non-idle.

[0050] The rate pool 502 is defined as the total number of resource slices minus the resource slices in use by the non-rate pool 504. The rate allocation of a volume is based on the individual volume fill rate and the total system fill rate. These inputs are then used to compute a rate allocation for a volume based on the available resources in the rate pool 502. The following invariants must be met as part of the rate allocation: (1) the allocation must be equal to or greater than the current resource consumption of the volume; (2) the allocation must be a multiple of the resource slice; and (3) the allocation must not result in the rate pool exceeding the resource size.

[0051] The non-rate pool 504 is initially empty. Non-rate allocation requests are received and the non-rate pool increases in size as slices are removed from the rate pool 502. The non-rate pool 504 shrinks when volumes return non-rate allocations to the rate pool. This occurs when a volume transitions to idle or when a volume begins to allocate from the rate pool. In the second case, non-rate resource slices stay with the volume, but they transition to the rate pool 502, thus shrinking the non-rate pool 504. The non-rate resource allocator is specifically designed to meet the resource needs of volumes that are currently incapable of using the rate allocation. Every non-idle volume can have at least one slice in its non-rate partition which serves as its basis for maintaining a valid resource fill rate (idle aggregates have 0 partition). When a volume transitions from idle to non-idle, the minimum partition is allocated to the volume from the non-rate pool 504. The minimum non-idle allocation will remain with the volume in the non-rate partition while the volume is non-idle. A volume can allocate resources from this pool 504 where the volume is idle; i.e., the volume has zero resource allocation for the resource in question. In this scenario the volume cannot produce a fill rate because it is not allowed to use a resource. In

order to provide a fill rate it must first be allowed to use the resource. In another example, a volume can allocate resources from this pool where the volume is not allowed to take a consistency point. In this scenario a volume may have resources in use and can provide a fill rate but since it cannot take a consistency point it must allocate additional resources beyond what has been given since it cannot release and reuse what has been allocated.

[0052] An exemplary method for providing dynamic consistency point triggers will now be described with reference to FIGS. 2-3 and 5. Referring more specifically to FIG. 6, an exemplary method for allocating resources to a requesting volume. In method 600, each volume receives a portion of dirty buffers based on rate based allocation and non-rate based allocation. The resource amount given to each volume then drives when a consistency point is triggered for the volume. In 602, resource slices are deducted from the rate pool. In step 604, the slices are allocated to a requesting volume. Resource allocation can occur upon the requests of additional resources at a volume (i.e. exhaustion of resources). Further, when a volume requests additional resources the other volumes can undergo resource reallocation based on the current known partition rates. This may result in an increase or decrease in rate partition for a volume depending on the current partition rates. An allocation may occur at the end of a consistency point due to a volume adjusting its partition based on the new drain rate. Further, an allocation may occur when a volume transitions from idle to non-idle. This will allocate the minimum partition to the volume from the non-rate pool. The minimum non-idle allocation will remain with the volume in the non-rate partition while the volume is non-idle.

[0053] In one embodiment, rate pool reallocation is restricted to occur once every 50 milliseconds. This is in place to limit unnecessary partition adjustments across volumes. Because the fill rate of the volumes is computed every 50 milliseconds, run rate reallocation will not occur unless the fill rates have changed, thus the 50 milliseconds check is in place. Rate pool allocation allocates multiple resource slices to a volume based on the volume to system partition rate ratio. Non-rate pool allocation allocates one slice to a volume at a time. In one embodiment, the purpose of an initial non-rate allocation is to allow the volume to generate a fill rate which can be used to allocate a rate partition. With a starting allocation of 2 times slice size, i.e. 256 buffers, the highest 50 millisecond rate is 5120 dirty buffers per second, which is large enough to get the volume started. This initial allocation must also be selected such that volumes with low load do not consume dirty buffer allocation unnecessarily. As this allocation increases the buffer space available for rate allocation decreases which can lead to performance penalties due to triggering more consistency points.

[0054] In step 606, a consistency point is triggered when half of the resource partition is consumed. In step 608, the non-rate resources allocated for a volume are returned to the rate pool 502 when the volume transitions to idle. In one embodiment, the volume transitions to idle if there has been no activity for the resource on a volume for 60 seconds, then the non-rate resource is returned. Thus, requiring the volume to repeat the non-rate process when it transitions to non-idle. The volume transitions to idle if the volume is capable of taking a consistency point. Because this is not detected immediately, the resources for the volume may stay in the non-rate pool until the volume completes a consistency point. When the resource transition occurs all non-rate resources except the non-idle minimum allocation are removed. The non-idle allocation remains until the aggregate is declared idle.

[0055] The dynamic consistency point triggers are designed to make a volume's consistency point duration more predictable and ensure all volumes have the resources needed to continue serving data. The consistency point time target is a constant maximum desired consistency point time. Meeting this time target is achieved by limiting the dirty buffers accepted in each consistency point.

[0056] With this time target defined a non-overloaded volume will conform to the time target thus preventing extremely long consistency points which may have unforeseen consequences (i.e. even when there are no dirty buffer nor NVLog suspends). In order to provide a soft time target guarantee, the consistency point is triggered before completely filling the time limited dirty buffer maximum. So the target consistency point time is actually half the defined maximum consistency point time target. In an exemplary embodiment, the default duration is thirty (30) seconds, which translates to a defined maximum time of sixty seconds (i.e. time when a resource is completely full). This target time is selected to prevent extreme modify operation outages and may be reduced if needed. The overhead of executing a consistency point and the metadata costs surrounding frequent smaller consistency points can be considered when selecting a time target. In some cases consistency points may actually run for 20-40 seconds without exhausting resources, then not be in consistency point for a few seconds. When the time target is reduced the behavior of this example volume can change such that it may now be in back-to-back consistency points. Another item to consider in selecting a time target is many smaller almost back-to-back consistency points may result in internal workloads such as delayed free, iron, truncate indirect, etc. (ones that need inodes not in consistency point or bitmaps in consistency point) not making progress. Further, another item to consider in selecting time target is the pacing targets a ten (10) second consistency point when no resources are being exhausted so the

aim is not to compete with pacing on the time front, but still provide much smaller outage windows.

[0057] In the back-to-back consistency point situation, the front-end workload's fill rate is greater than the consistency point's drain rate. If the fill rate is not throttled by admission control, the back-to-back consistency points are unavoidable and overloaded front-end messages will suspend during consistency point. The suspension time is observed as no-write-op window. In the back-to-back consistency point case, by reducing time to a smaller, fixed time, a long consistency point is broken into multiple smaller consistency points and the no-write-op window is shortened.

[0058] The time restricted dirty buffer limit strives to limit consistency point time to some duration of time. This is achieved through measurement of back-end write performance, and then by limiting the amount of work given to a consistency point such that it can be completed in that time frame. The work given to a consistency point is a combination of client write and internal write operations. The write workload can be defined as dirty buffers for consistency point to clean.

[0059] Internal workloads that generate work for consistency points are dynamically detected and the amount of work each is allowed to generate is restricted. When the storage server attempts to dirty an in-memory buffer, a set of rules is applied to the message, which determines if the running message is an internal workload message. If the criterion is met the message type is labeled as an internal workload message type. This information is then used when processing new messages to quickly identify an internal workload message. Each buffer dirtied by an internal workload message is accounted for in order to stop new work from being generated when a defined limit is reached, or once the consistency point completes.

[0060] The present disclosure is not limited to the precise construction and compositions disclosed herein; any and all modifications, changes, and variations apparent from the foregoing descriptions are within the spirit and scope of the disclosure as defined in the appended claims. Moreover, the present concepts expressly include any and all combinations and sub combinations of the preceding elements and aspects.

CLAIMS

What is claimed:

1. A method for providing dynamic consistency point triggers by dividing resources between volumes within a storage system, the method comprising:

determining a volume's capacity to utilize resources based on the volume's performance;

receiving an allocation of the divided resources based on total system resources available within the storage system and the volume's performance; and

triggering a consistency point upon exhausting a threshold percentage of the received/allocated resources.

2. The method of claim 1, further comprising restricting an internal workload within the volume when performing the consistency point.

3. The method of claim 1 or 2, further comprising restricting an internal workload upon exhausting a threshold percentage of the received/allocated resources.

4. The method of any preceding claim, wherein determining the volume's capacity comprises determining a contribution to a fill rate of the storage system via determining a resource partition, wherein the resource partition is a sum of a rate partition and a non-rate partition of the volume.

5. The method of any preceding claim, wherein determining the volume's capacity comprises determining a rate based allocation and a non-rate based allocation of the volume, wherein the rate based allocation and the non-rate based allocation vary.

6. The method of claim 5, wherein the rate based allocation and the non-rate based allocation vary every 50 milliseconds or less.

7. A computing device, comprising:

a memory containing machine readable medium comprising machine executable code having stored thereon instructions for performing a method of providing dynamic consistency point triggers by dividing resources between volumes within a storage system;

a processor coupled to the memory, the processor configured to execute the machine executable code to cause the processor to perform the method of any preceding claim.

8. A computing device, comprising:

a memory containing machine readable medium comprising machine executable code having stored thereon instructions for performing a method of providing dynamic consistency point triggers by dividing resources between volumes within a storage server;

a processor coupled to the memory, the processor configured to execute the machine executable code to cause the processor to:

determine a volume's capacity to utilize the resources based on the volume's performance;

receive an allocation of the divided resources based on total system resources available within the storage server and the volume's performance; and

trigger a consistency point upon exhausting a threshold percentage of the received/allocated resources.

9. The computing device of claim 8, further comprising restricting an internal workload within the volume when performing the consistency point.

10. The computing device of claim 8 or 9, further comprising restricting an internal workload upon exhausting a threshold percentage of the received/allocated resources.

11. The computing device of any of claims 8 to 10, wherein determining the volume's capacity comprises determining a contribution to a fill rate of the storage server via determining a resource partition, wherein the resource partition is a sum of a rate partition and a non-rate partition of the volume.

12. The computing device of any of claims 8 to 11, wherein determining the volume's capacity comprises determining a rate based allocation and a non-rate based allocation of the volume, wherein the rate based allocation and the non-rate based allocation vary.

13. The computing device of claim 12, wherein the rate based allocation and the non-rate based allocation vary every 50 milliseconds or less.

14. A non-transitory machine-readable medium having stored thereon instructions for performing a method of providing dynamic consistency point triggers by dividing resources between volumes within a storage system comprising machine executable code which when executed by at least one machine, causes the machine to perform the method of any of claims 1 to 6.

15. A non-transitory machine-readable medium having stored thereon instructions for performing a method of providing dynamic consistency point triggers by dividing resources between volumes within a storage server comprising machine executable code which when executed by at least one machine, causes the machine to:

determine a volume's capacity to utilize the resources based on the volume's performance;

receive an allocation of the divided resources based on total system resources available within the storage server and the volume's performance; and

trigger a consistency point upon exhausting a threshold percentage of the received/allocated resources.

16. The non-transitory computer readable medium of claim 15, further comprising restricting an internal workload within the volume when performing the consistency point.

17. The non-transitory computer readable medium of claim 15 or16, further comprising restricting an internal workload upon exhausting a threshold percentage of the received/allocated resources.

18. The non-transitory computer readable medium of any of claims 15 to 17, wherein determining the volume's capacity comprises determining a contribution to the fill rate of the storage server via determining a resource partition, wherein the resource partition is a sum of a rate partition and a non-rate partition of the volume.

19. The non-transitory computer readable medium of any of claim 15 to 18, wherein determining the volume's capacity comprises determining a rate based allocation and a non-rate based allocation of the volume, wherein the rate based allocation and the non-rate based allocation vary.

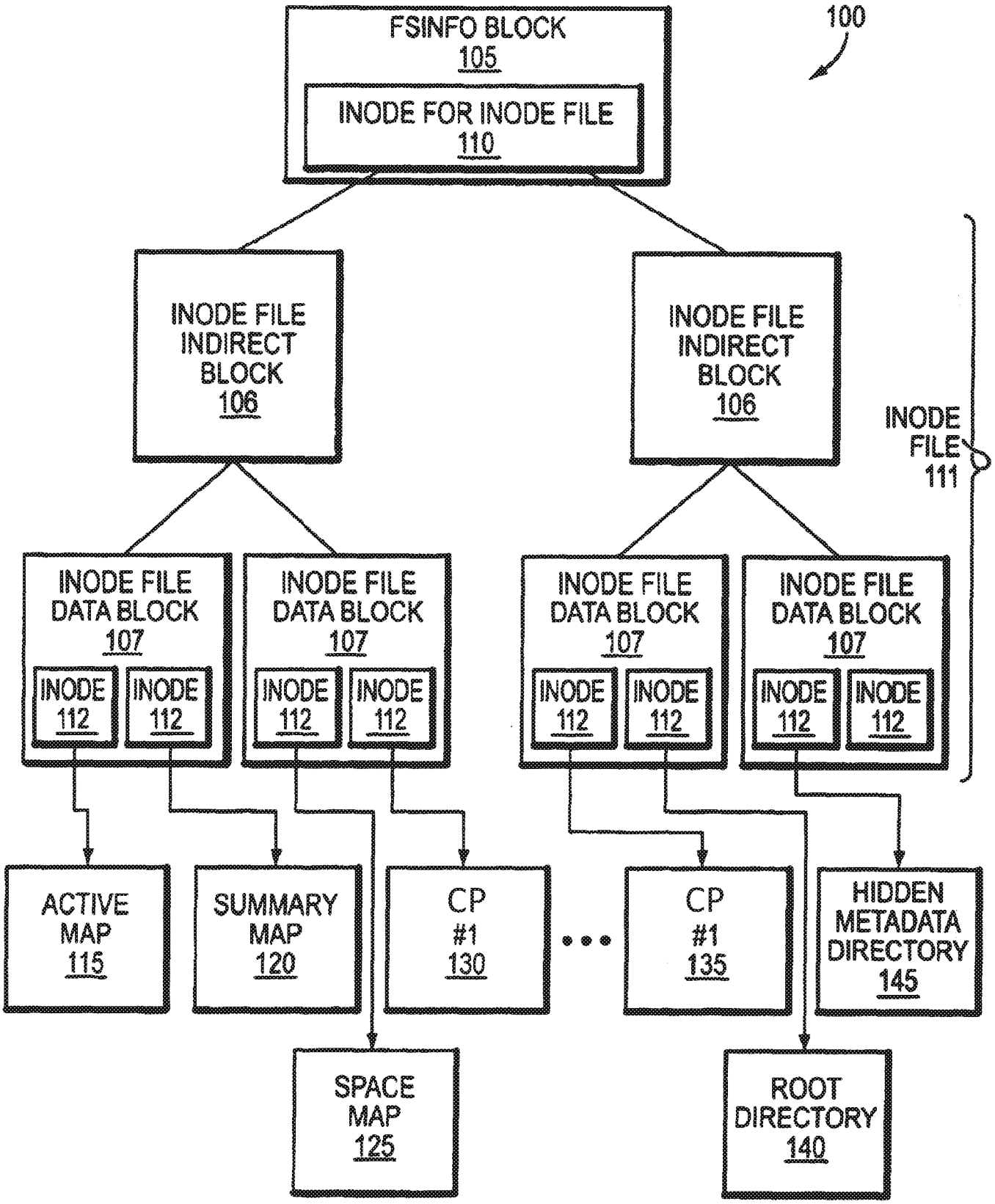

100
FSINFO BLOCK
105
INODE FOR INODE FILE
110
INODE FILE
INDIRECT
BLOCK
106
INODE FILE
INDIRECT
BLOCK
106
INODE
FILE
111
INODE FILE
DATA BLOCK
107
INODE
112
INODE
112
INODE FILE
DATA BLOCK
107
INODE
112
INODE
112
INODE FILE
DATA BLOCK
107
INODE
112
INODE
112
INODE FILE
DATA BLOCK
107
INODE
112
INODE
112
ACTIVE
MAP
115
SUMMARY
MAP
120
CP
#1
130
CP
#1
135
HIDDEN
METADATA
DIRECTORY
145
SPACE
MAP
125
ROOT
DIRECTORY
140

FIG. 1

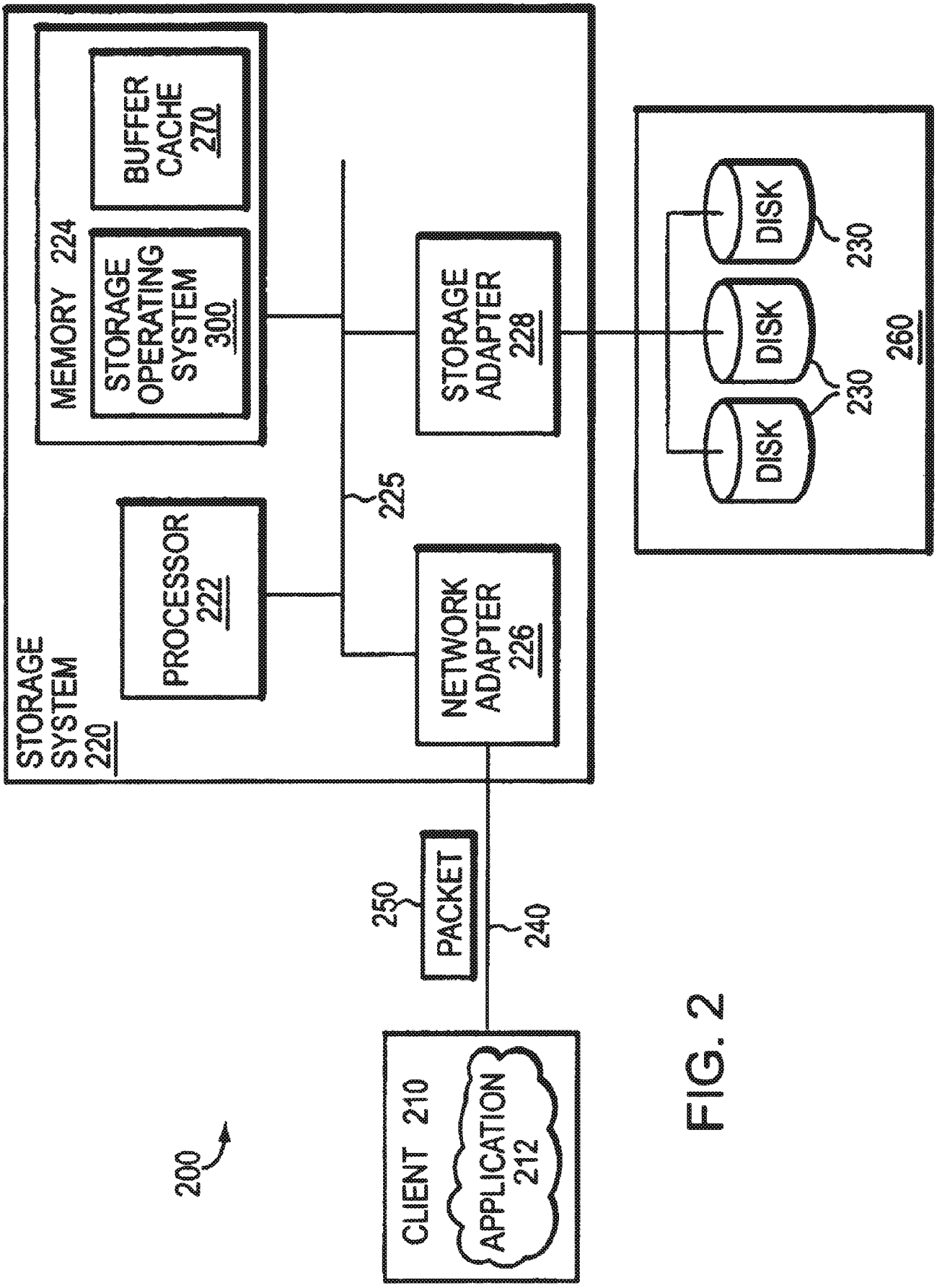

200
CLIENT 210
APPLICATION 212
250
PACKET
240
STORAGE SYSTEM 220
PROCESSOR 222
225
NETWORK ADAPTER 226
STORAGE ADAPTER 228
MEMORY 224
STORAGE OPERATING SYSTEM 300
BUFFER CACHE 270
DISK
DISK
DISK
230
230
260

FIG. 2

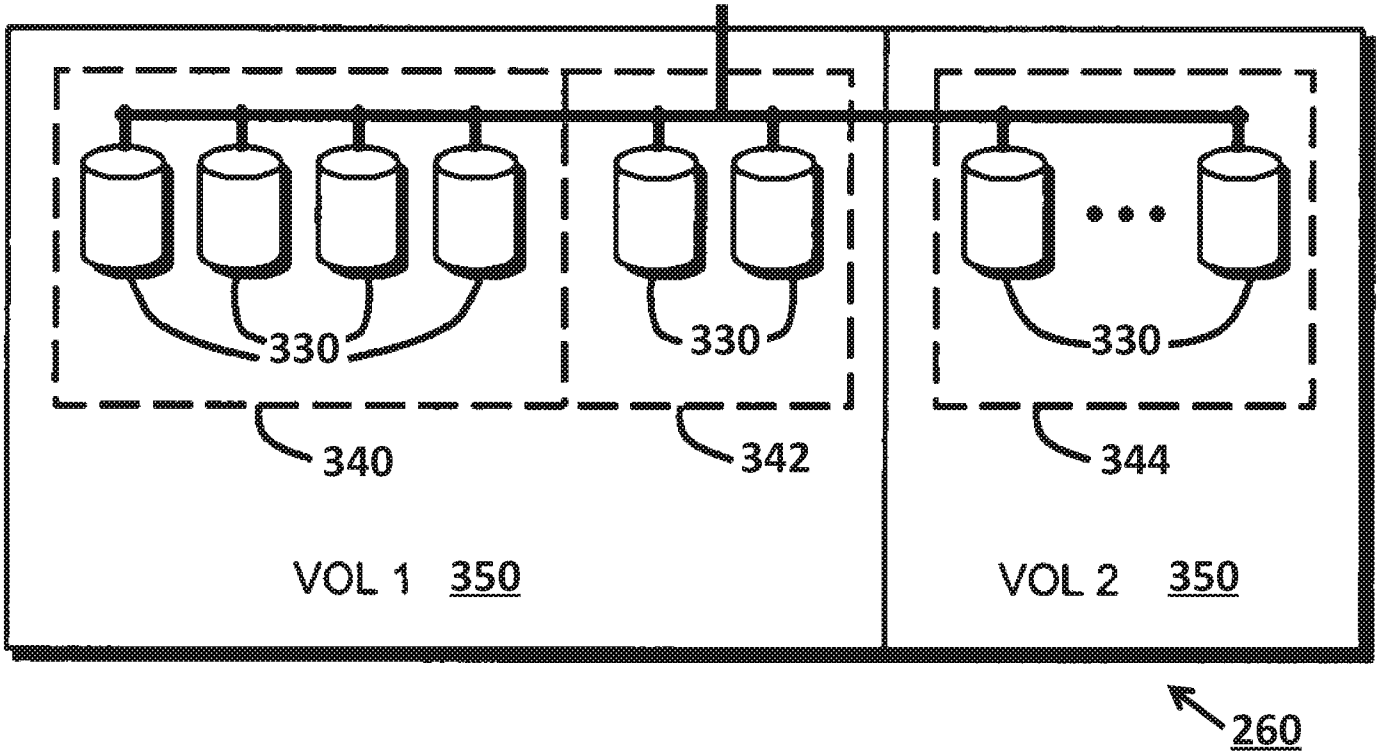

330
330
330
340
342
344
VOL 1 350
VOL 2 350
260

FIG. 3

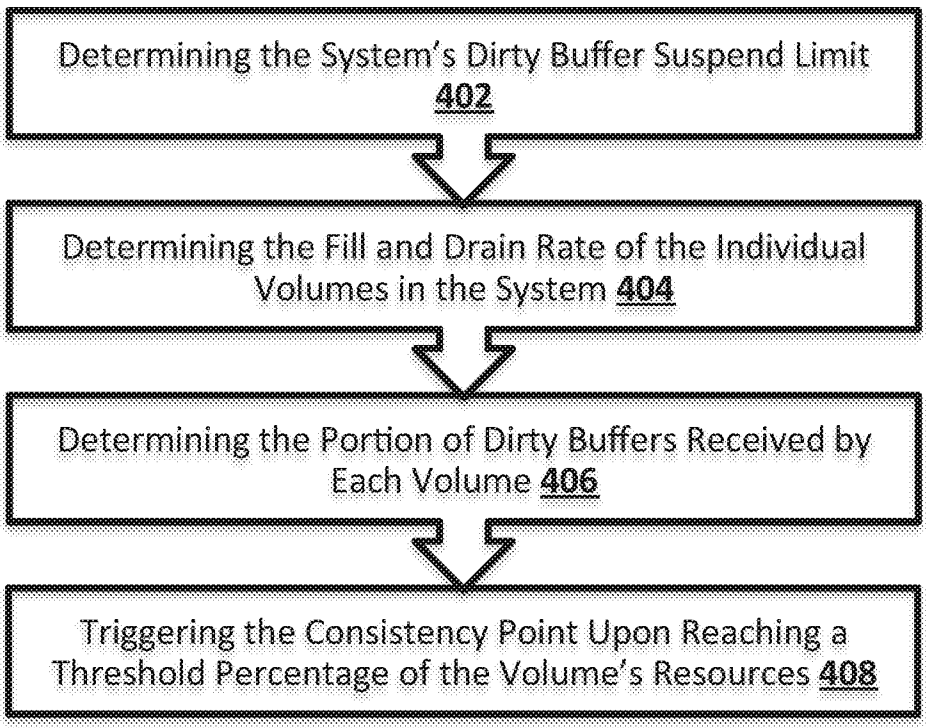

Determining the System's Dirty Buffer Suspend Limit 402
Determining the Fill and Drain Rate of the Individual Volumes in the System 404
Determining the Portion of Dirty Buffers Received by Each Volume 406
Triggering the Consistency Point Upon Reaching a Threshold Percentage of the Volume's Resources 408

FIG. 4

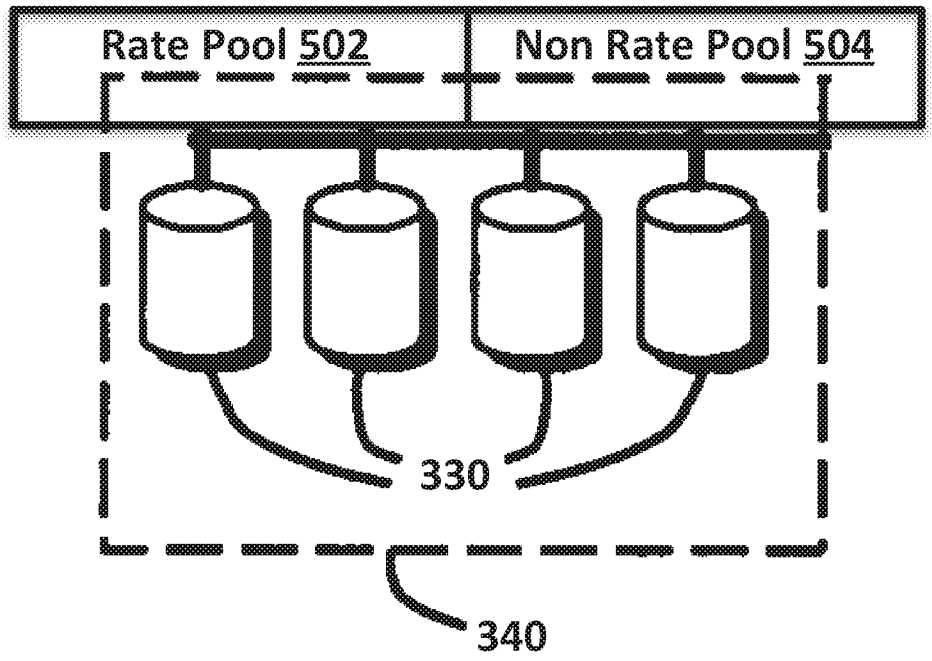

Rate Pool 502
Non Rate Pool 504
330
340

FIG. 5

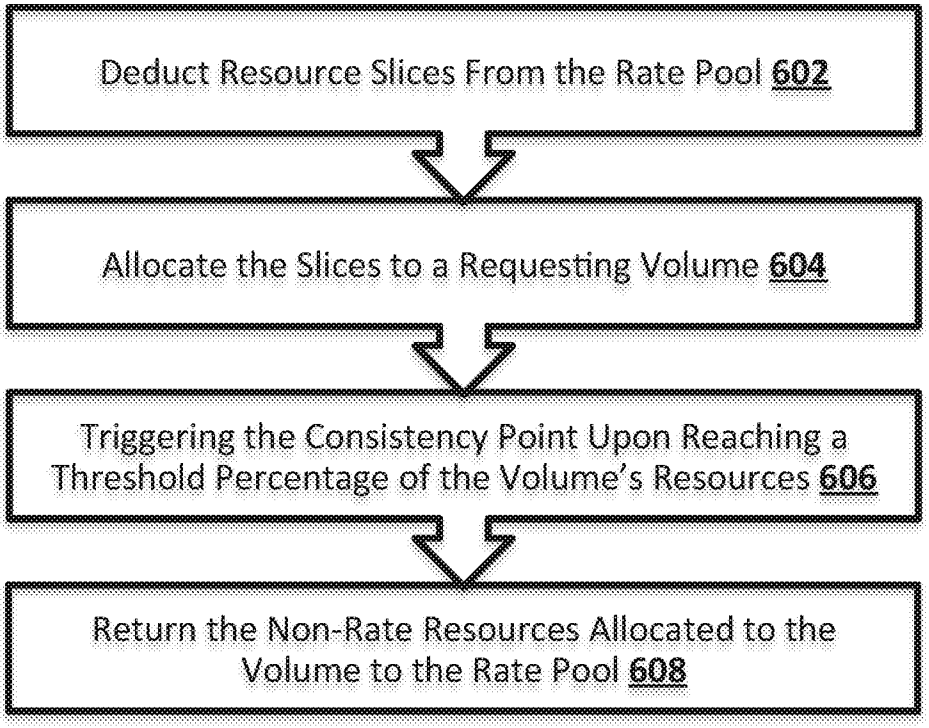

Deduct Resource Slices From the Rate Pool 602
Allocate the Slices to a Requesting Volume 604
Triggering the Consistency Point Upon Reaching a Threshold Percentage of the Volume's Resources 606
Return the Non-Rate Resources Allocated to the Volume to the Rate Pool 608

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2017/019067

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F3/06
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6 289 356 B1 (HITZ DAVID [US] ET AL) 11 September 2001 (2001-09-11) the whole document -----	1-19

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 May 2017	31/05/2017

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Andlauer, J

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2017/019067

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6289356 B1	11-09-2001	US 5963962 A	05-10-1999
		US 6289356 B1	11-09-2001
		US 2002091670 A1	11-07-2002
		US 2004260673 A1	23-12-2004